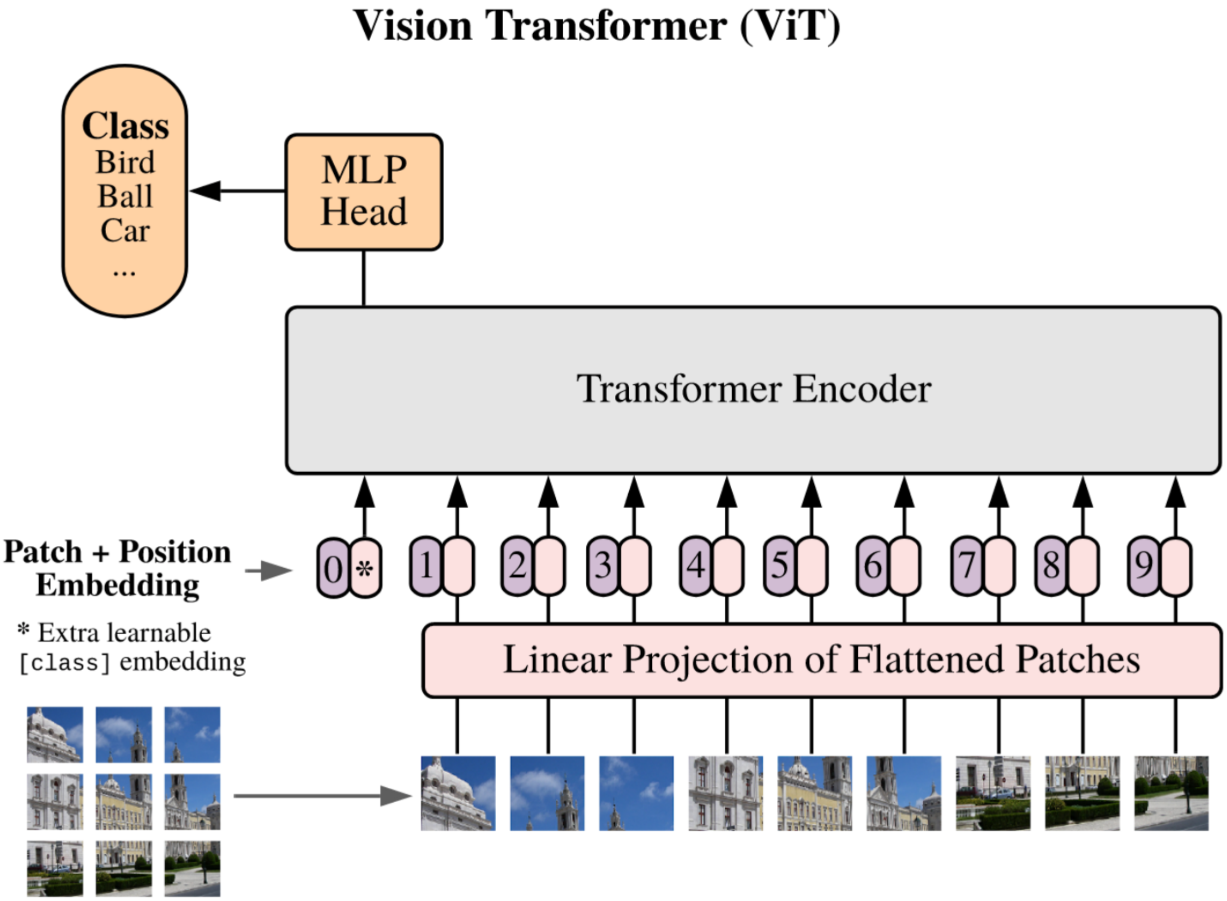
Transformers

Here is the general architecture of the ViT model

The architecture includes:

- A step for splitting the image into disjoint patches with a linear projection of the patches
- Positional encoding to capture the absolute spatial information of each patch in the image
- The addition of a "[CLS]" token, initialized randomly and learned during training
- A transformer encoder that learns the representation of each token
- A classification head that performs class prediction from the re-embedded [CLS] token at the output of the encoder


Vision Transformer (ViT)
Class
Bird
Ball
Car
...
MLP Head
Transformer Encoder
Patch + Position Embedding
* Extra learnable [class] embedding
0 *
1
2
3
4
5
6
7
8
9
Linear Projection of Flattened Patches

Linear Projection of flattened patches

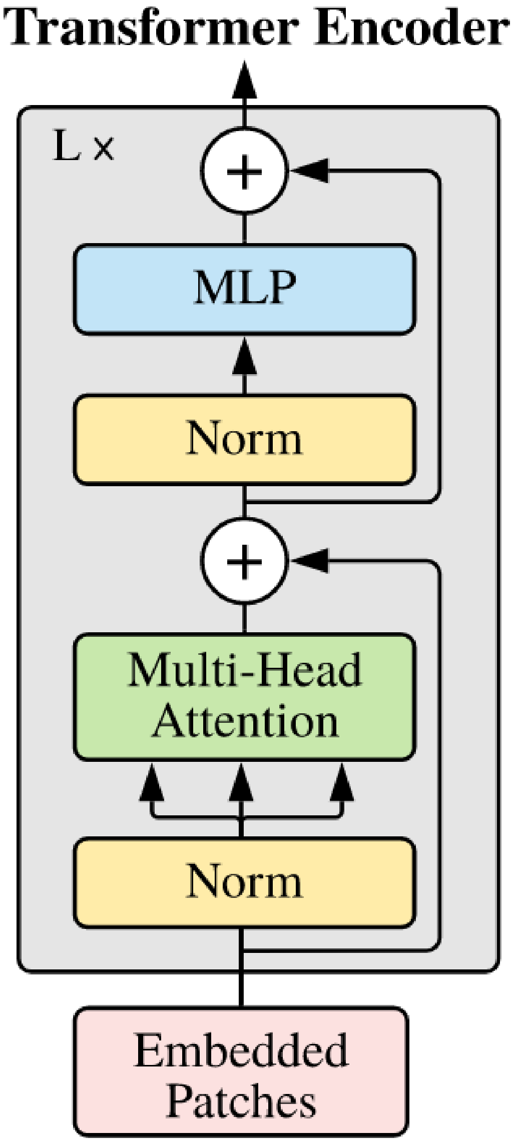
Q1: First, we need to code the patch embedding. In NLP, it could be using a word2vec embeddings, but we need something different in Computer Vision.

You are going to use a specific 2d convolution to play the role of the so-called "Linear Projection" on the image above and process each patch independently without overlap. Think a bit how you could do that.

Then, you need to manipulate the dimensions of the resulting tensor to get a final tensor of shape `(batch_size, nb_tokens, embedding_dim)`. *Hint: You will need a `view/reshape` and `transpose/permute` operators.*

As usual, remember to check the output shape of your layer by trying with a random tensor!

Transformer encoder



Q2: MLP

Now we need to build a transformer block. Let's first build the easiest part, the MLP! By now, you should be confident how to code one.

Code a MLP with one hidden layer. Note that the input and output dimensions will be the same. Instead of ReLU, use the activation GELU, which is a slight alternative often used for transformers.

Q3: Self-attention

Now we are going to build the famous **Self-Attention**.

- What is the main feature of self-attention, especially compared to its convolutional counterpart. What is its main challenge in terms of computation/memory?

$$Q = X W_Q \in \mathbb{R}^{T \times d_K} \text{ (Query)}$$

$$K = X W_K \in \mathbb{R}^{T \times d_K} \text{ (Key)}$$

$$V = X W_V \in \mathbb{R}^{T \times d_V} \text{ (Value)}$$

$$T : \text{sequence length}$$

$$d_K, d_V : \text{hidden (embedding) dimension of queries/keys/ values, here: } d_K = d_V$$

- At first, we are going to only consider the simple case of one head. Write the equations and complete the following code. And don't forget a final linear projection at the end!

Q4 : Multi-head self-attention

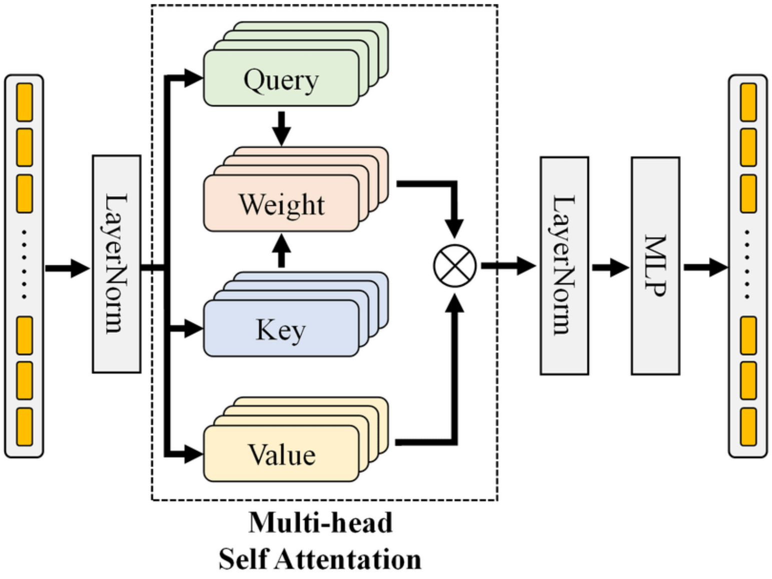
Now we need to build a Multi-Heads Self-Attention.

- Write the equations and complete the following code to build a Multi-Heads Self-Attention.

$$\text{MultiHeadAttention}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^0$$

$$\text{head}_i = \text{self-attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Vizualization of Multi-Head Attention Layer



- The total embedding size will be shared equally among all heads.

Q5: Transfomer block

Now, we need to build a Transformer **Block** as described in the image below.

- Write the equations and complete the following code.

- For the Layer Normalization, use PyTorch LayerNorm: https://pytorch.org/docs/stable/generated/torch.nn.LayerNorm.html

Q6: Full ViT model

Now you need to build a ViT model based on what you coded in the previous questions. There are additional components that should be coded such as the Class token, Positional embedding and the classification head.

a) Explain what is a class token and why we use it?
➤ Initialized randomly, will be trained during traning by backprob, weight vector, it is re-embedded again and again through the transformer network.
➤ Comes in at the end for the classification. Information accumulates in the class token. Idea comes from Bert (NLP model)
➤ In this setup, the MLP works only on the learned class tokens.
b) Explain what is the the positional embedding and why it is important?
➤ Multi-head attention layer is permutation equivariant (does not change with changed word/patch order). Therefore, we have to pass additional clues to the model to respect the word order within a sentence.
We use it so as not to loose the information about the spatial position of the patches
➤ include graph of sinus/cosinus for positional embedding
c) Test different hyperparameters and explain how they affect the performance. In particular embed_dim, patch_size, and nb_blocks.

For PE, you can use a sinusoidal encoding (see below), or fully learned.

Rewrite positional encoding formula

$$\mathrm{PE}(k,i) = \begin{cases} \sin\left(\frac{\text{position}}{10000^{\frac{i}{d}}}\right) \text{ if } i \text{ even} \\ \cos\left(\frac{\text{position}}{10000^{\frac{(i-1)}{d}}}\right) \text{ else} \end{cases} \Leftrightarrow \mathrm{PE}(k,2i) = \sin\left(\frac{\text{position}}{10000^{\frac{2i}{d}}}\right) \mathrm{PE}(k,2i+1) = \cos\left(\frac{\text{position}}{10000^{\frac{2i}{d}}}\right)$$

Q7: Experiment on MNIST!

Experimental analysis

- Test different hyperparameters and explain how they affect the performance. In particular embed_dim, patch_size, and nb_blocks.
- Comment and discuss the final performance that you get. How to improve it?
- What is the complexity of the transformer in terms of number of tokens? How you can improve it?

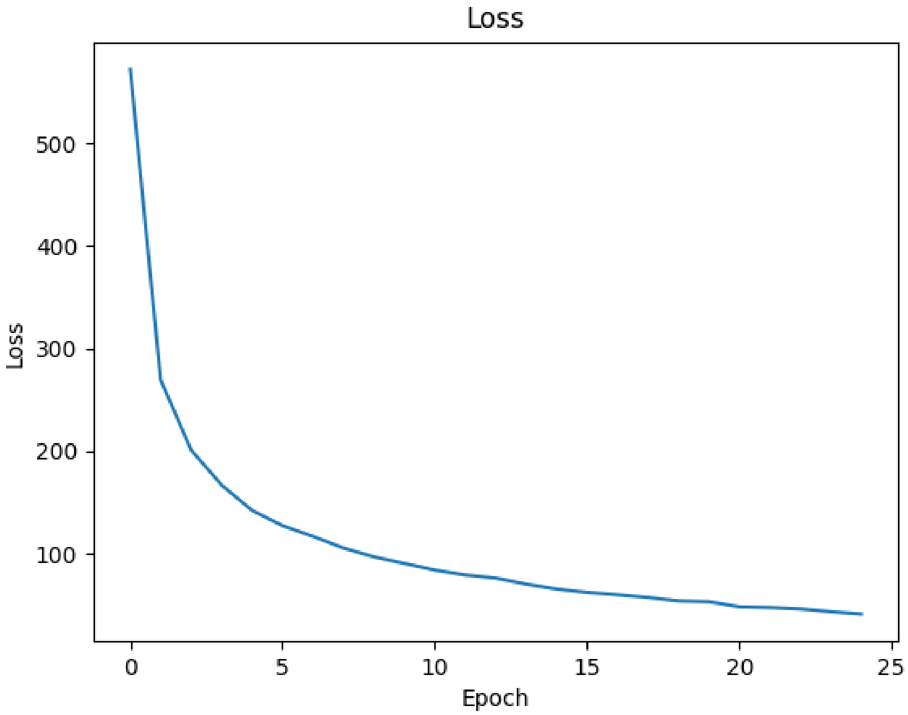
Q8: larger transformers

Try to use bigger transformer, for example the ViT-S from the timm library. Test with and without intialization from imagenet.

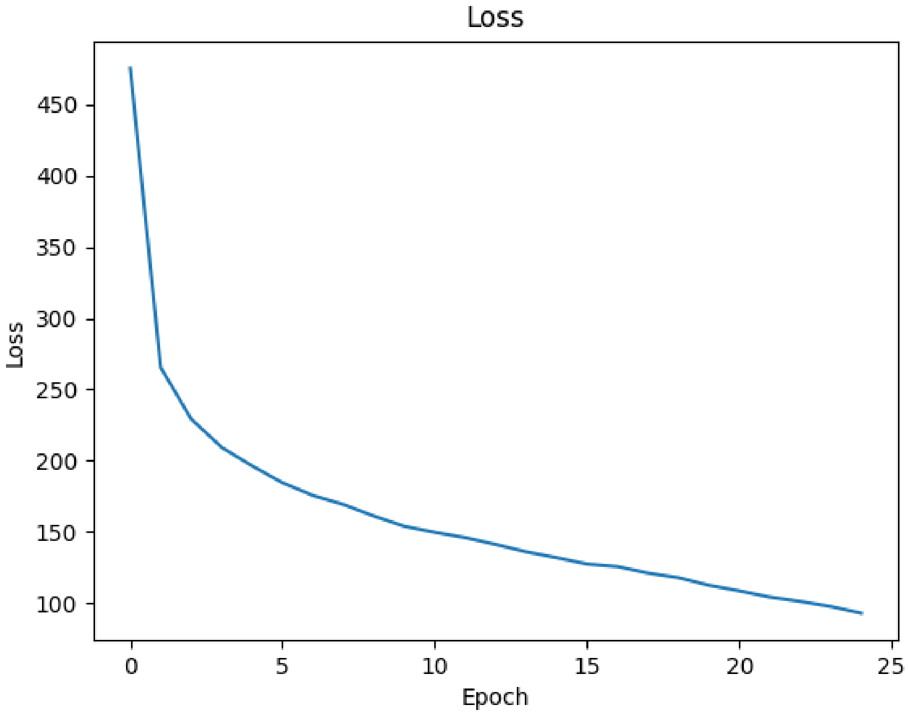
a) Load the model using the timm library without pretrained wewights. Try to apply it directly on a tensor with the same MNIST images resolution. What it is the problem and why we have it? Explain if we have also such problem with CNNs. As ViT takes RGB images, the input tensor should have 3 channels.

→ Input shape is fixed for CNNs and ViTs by choice of architecture. We can go around by rezising (shrinking or interpolating) but we are not ideally using either the image details or the efficiency of the CNN. (same for channels)

b) There is a trcik in timm that allows to use pretrained models with different image resolution and number of classes. Try to reload the model to be able to use on MNIST images:
c) redo the training with the pretrained ViT-S

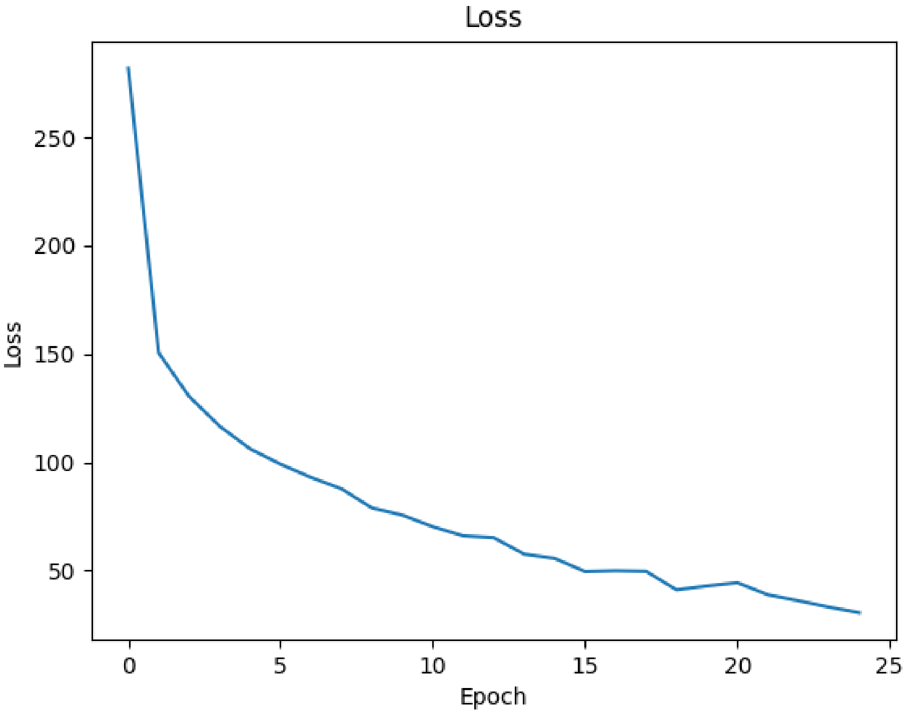


c) redo the training but with the ViT-S pretrained on ImageNet



d) Comment the final results and provide some ideas on how to make transformer work on small datasets. You can take inspiration from some recent work.

--> Transformers as many NNs are overparametrized for small datasets and we have to rely on regularization techniques (e.g. early stopping, dropout,...) or increase the number of training samples (e.g. training augmentations, pre-training on a different dataset and finetuning afterwards (transfer learning),...)


Loss
Loss
Epoch